

NPS-EC-02-003

NAVAL POSTGRADUATE SCHOOL

Monterey, California



Signal to Noise Ratio Improvement Using Wavelet and Frequency Domain Based Processing

by

R. Hippenstiel

May 24, 2002

Approved for public release; distribution is unlimited.

Sponsored by the Naval Postgraduate School Center for Reconnaissance Research.

20020730 166

NAVAL POSTGRADUATE SCHOOL
Monterey, California

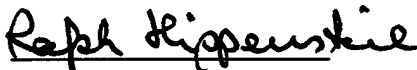
RADM David R. Ellison
Superintendent

R. Elster
Provost

This report was sponsored by the Naval Postgraduate School Center for
Reconnaissance Research.

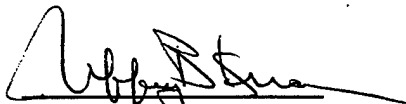
Approved for public release; distribution is unlimited.

The report was prepared by:



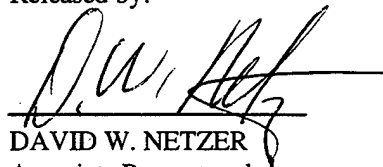
RALPH HIPPENSTIEL
Associate Professor
Department of Electrical and
Computer Engineering

Reviewed by:



JEFFREY B. KNORR
Chairman
Department of Electrical and
Computer Engineering

Released by:



DAVID W. NETZER
Associate Provost and
Dean of Research

REPORT DOCUMENTATION PAGE			Form Approved OMB No. 0704-0188	
<small>Public reporting burden for the collection of information is estimated to average 1 hour per response, including the time for reviewing instructions, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing this burden to Washington Headquarters Services, Directorate for Information Operations and Reports, 1215 Jefferson Davis Highway, Suite 1204, Arlington VA 22202-4302, and to the Office of Management and Budget, Paperwork Reduction Project (0704-0188), Washington DC 20503.</small>				
1. AGENCY USE ONLY (Leave blank)	2. REPORT DATE May 24, 2002	3. REPORT TYPE AND DATES COVERED Final Report, March-December 2001		
4. TITLE AND SUBTITLE Signal to Noise Ratio Improvement Using Wavelet and Frequency Domain Based Processing		5. FUNDING NUMBERS MTPR NO. B448212		
6. AUTHOR(S) R. Hippenstiel		8. PERFORMING ORGANIZATION REPORT NUMBER NPS-EC-02-003		
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) Department of Electrical and Computer Engineering Naval Postgraduate School Monterey, CA 93943-5000		10. SPONSORING/MONITORING AGENCY REPORT NUMBER		
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES) Center for Reconnaissance Research Naval Postgraduate School Monterey, CA 93943		11. SUPPLEMENTARY NOTES The views expressed in this report are those of the author and do not reflect the official policy or position of the Department of Defense or the United States Government.		
12a. DISTRIBUTION/AVAILABILITY STATEMENT Approved for public release; distribution is unlimited.		12b. DISTRIBUTION CODE A		
13. ABSTRACT (Maximum 200 words) This work investigates the use of wavelet and FFT based decompositions to improve the signal to noise ratio of noisy signals. In their respective transform domains, median filtering or predictive filtering is employed. Prior to the decompositions a short time domain median filter is used. As a benchmark, only a median time domain filter (order 3) is used and for part of the work the pre-filtering is disabled. Three test signals are used: two frequency chirped signals and a Barker coded BPSK signal. The most effective processing sequence for the chirp signals is median filtering, followed by FFT processing, which in turn, is followed by median filtering of the FFT transform coefficients. For the BPSK signal, the time domain median filter provided the best results.				
14. SUBJECT TERMS denoising, time-frequency/scale distributions, electronic warfare			15. NUMBER OF PAGES 55	
			16. PRICE CODE	
17. SECURITY CLASSIFICATION OF REPORT UNCLASSIFIED	18. SECURITY CLASSIFICATION OF THIS PAGE UNCLASSIFIED	19. SECURITY CLASSIFICATION OF ABSTRACT UNCLASSIFIED	20. LIMITATION OF ABSTRACT UL	

Table of Contents

1. Introduction	1
A) Background	1
B) Organization	2
2. Decomposition	2
3. Wavelet Based Processing	4
i) Median Filtering	6
ii) Predictive Filtering	6
4. FFT Based Processing	7
5. Data and Processor Parameters	8
6. Simulation Results	9
7. Conclusion	17
8. References	18
Appendix A	19
Appendix B	

Executive Summary

This work investigates the use of wavelet and FFT decompositions in the context of denoising signals embedded in white Gaussian noise. The data is pre-filtered (3rd order median filter) in part of the work and is not pre-filtered in another part. Once the noisy signal is decomposed (using the wavelet or the FFT), the transform coefficients are denoised using a predictive filter (order 2) or another median filter (order 3). Three test signals are used: a frequency chirp with constant amplitude, a frequency chirp with RC time constant type amplitude modulation, and a Barker coded BPSK signal. Pre-filtering, coupled with FFT processing and follow on median filtering provides the best MSE results for the two chirped waveforms. The decomposition approach does not work for the BPSK signal. In this case, a simple median filter, employed in the time domain, is shown to be a better denoising candidate.

SIGNAL TO NOISE RATIO IMPROVEMENT USING WAVELET AND FREQUENCY DOMAIN BASED PROCESSING

Ralph Hippenstiel
Electrical and Computer Engineering Department
Naval Postgraduate School
Monterey, Ca 93943

1. Introduction

A. BACKGROUND

In the majority of applications only a noise-corrupted version of the signal of interest is available. In many problems it is desirable to enhance the signal to allow a more precise determination of the signal parameters, such as duration, chip rate, chirp rate, modulation type, carrier frequency, etc. The enhancement concept can also be applied to improve time delay based localization. In time delay estimation both channels are denoised. The cross correlation properties of the signal channels are also used to obtain a precise time difference of arrival (TDOA) estimate [1].

Wavelet (WL) decomposition is used in many signal processing applications. One important application is noise reduction, also called denoising. Each transform coefficient represents a measure of the correlation between the signal and a WL basis function. Large coefficients represent good correlation, while small coefficients represent poor correlation. Denoising tends to retain the coefficients that preserve the signal and remove the coefficients that represent noise. The difficult part of the denoising process is to decide which components to emphasize and which ones to de-emphasize. One can also replace the WL decomposition with a Fourier type decomposition, so that in the time-

frequency domain the signal related components are emphasized. Again the difficulty is to decide which components are signal related and which ones are not.

The idea behind signal enhancement is simple: transform the noisy signal to the time-scale or time-frequency domain (i.e., analysis), reduce the noise effects, and transform the modified coefficients back to the time domain (i.e., synthesis). Since the signal will not occupy all frequency regions at all times, some of the noise can be removed. The noise reduction works best when the signal is concentrated in time, or in frequency, or both in frequency and time.

B. ORGANIZATION

This report examines two transform techniques that are used to denoise (or filter) the transform coefficients in their respective transform domains. A median filtering and predictive filtering method is used to denoise the data in the transform domain. A pre-filtering approach (using a median filter in the original time domain) is also implemented and compared to results when no pre-filtering is attempted.

2. DECOMPOSITIONS

The basic idea behind denoising (or filtering) is to separate the noisy signal into its constituent components. That is, separation into parts primarily associated with the signal components and those that are not. The noise removal tends to retain the signal related components and remove as much as is possible the components that relate to the noise only. Inadvertently, some signal components are removed while some noise components are retained. To allow separation of noise and signal, the noisy signal is

mapped into one of the two transform domains, denoised, and mapped back to the original time domain. The mappings considered are decompositions based on the wavelet (WL) and the fast Fourier transform (FFT). The WL and FFT operations are shown in block diagram form in figures 1 and 2, respectively. The symbols s and u are used to denote the signal and the noise, respectively.

The WL transform allows a decomposition into spectral bands that are proportional to the band center of the spectral region, i.e., constant Q-filtering. For all detail and approximation coefficients, the sample rate is kept at the Nyquist rate, that is, every scale (i.e., every band pass filter) has an output data rate matched to the bandwidth of the filter [2].

On the other hand, the FFT transform provides for a uniformly spaced filter bank having a constant band width a constant data output rate. The data rate can be matched to the Nyquist rate. For practical reasons, overlap processing is used, which makes the output data rate larger than the minimum rate (i.e., overlap is 75 percent, the output rate is 4 times the Nyquist rate) [2].

Block Diagram Using Wavelet Denoising

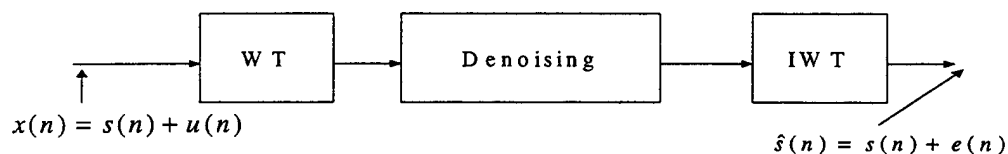


Figure 1: System block diagram for WL based denoising.

Block Diagram Using FFT Based Denoising

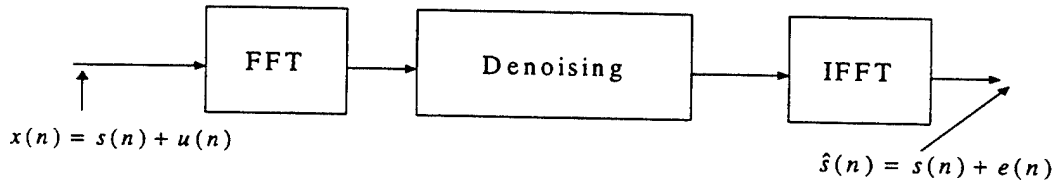


Figure 2: System block diagram for FFT based denoising.

Prior to the decomposition a pre-filter can be used in the denoising process. If pre-filtering is invoked then it is accomplished using a 3rd order median filter.

3. WAVELET BASED PROCESSING

Wavelet (WL) processing is also known as constant Q-filtering, pro-proportional band width processing, multi-rate filtering, and time-scale processing. Figure 3 shows some of the details of the WL based processing and the weighting of the detail and approximation coefficients. In each region of interest a low and high pass filter is used to edit out signals occupying the respective spectral band region.

The denoising procedure consists of three steps. These steps are discrete wavelet decomposition, scaling of each subband sequence, and an inverse wavelet transform. The modified subband sequences are obtained by weighting each subband as given by

$$d_i^c = w_{d_i} d_i ; \quad \text{and}$$

$$a_i^c = w_{a_i} a_i ; \quad i = 1, 2, \dots, J ,$$

where w_{d_i} and w_{a_i} represent the weighting terms.

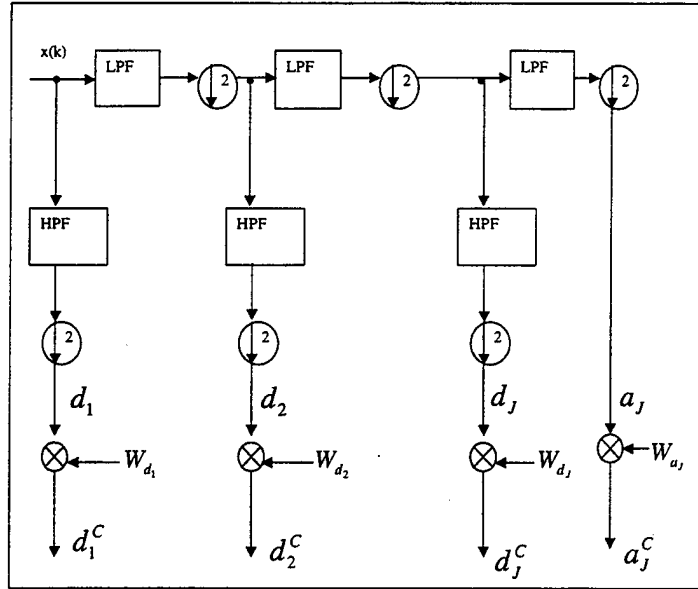


Figure 3: Processing in the wavelet domain.

The weighting accomplishes the signal enhancement by de-emphasizing noise related components. The weighting schemes used in this report is median filtering and predictive filtering. Classical denoising schemes as advocated in Donahoe [3-5] fail at low SNR (i.e., in the negative dB region) and are not investigated in this report. Some detail on the classical denoising performance in conjunction with GSM signal localization can be found in [6].

Transform coefficients denoising is obtained using a median filter or an optimal (Wiener) predictor.

i) Median filtering:

The median filter is applied to each scale output. The detail (band pass) outputs are denoted by d_i (for $i = 1, \dots, J$), while the single approximation (low pass) output is referred to as a_J . Each output (detail and approximation) is median filtered to de-emphasize (reduce) the band limited white noise contribution. The median filter takes several sequential data points and uses as the filtered output the data point that is obtained as the middle point when ranking is invoked. A median filter, of length 3, is applied to the first J sequences of detail coefficients and the J^{th} sequence of the approximation coefficients of the WL transform. The median filter replaces the center point of the window with the median value of all the points contained in the window. Ranking the values and selecting the central value achieves this. The length of the window is very important. For example, for a narrowband signal a long window length maybe appropriate. If the signal is non-stationary, a short window tends to be better. If one does not have a priori information about the source signal this can be a drawback. Based on empirical evidence a median filter of size 3 was selected for the three data sets used in the simulations.

ii) Predictive Filtering:

The predictive filtering [7] is accomplished using a Wiener FIR filter of size 2. The predictive filter predicts the predictable part that is thought to be the signal, hence its output tends to have little residual noise. A predictive filter is applied to all generated detail sequences and the final approximation sequence of interest. Each filter output has the same number of data samples as its corresponding input sequence. During the

initialization, (i.e., the transitory region, i.e., the first two data output points), the input (unfiltered) data is used as the estimate of the signal component. The second order predictor works best in that it produced a smaller MSE compared to predictors of different sizes.

4. FFT BASED PROCESSING

In the FFT based approach, the constant Q-filters are replaced with constant bandwidth filters (i.e., with the FFT bins). As in the WL based approach, two denoising (filtering) approaches are taken: median filtering and predictive filtering. All frequency bins have the same number of data points (i.e., have the same sampling rate). The symbol r is used to denote the down and up sampling. The down sampling is a function of the overlap factor and the data length used in the FFT transform.

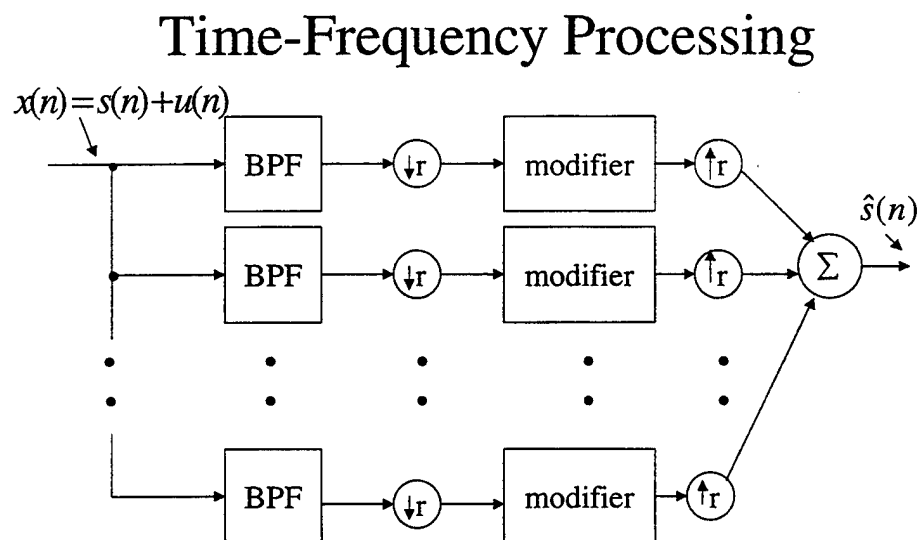


Figure 4: Schematic data flow in time-frequency (FFT) based denoising.

5. DATA AND PROCESSOR PARAMETERS

Three test signals are used in the simulations. All simulations are conducted using Matlab (version 6.0) [8]. The first one is a chirp signal that increases in amplitude from zero to a maximum in an RC time constant fashion. This signal is referred to as the Doppler signal in reference [9]. The second signal is a linearly chirped sinusoid that has a constant amplitude. The third signal is a BPSK signal that uses a Barker code of length 5 [10], for each one of its information bits.

The data length is fixed at 512 samples. The power of the signals is normalized to be unity. The additive Gaussian noise is white having a variance that is adjusted to obtain the desired signal-to-noise ratio (SNR). The following SNR values are used:

-10, -6, -3, 0, 3, and 6 decibels (dB). To ensure statistical reliability, each signal is processed 100 times using independent noise realizations. The third signal uses a random start time, that is, a delay uniformly distributed between 0 and 27 data points. Its carrier frequency is selected so that there are 9 data points per period of the carrier. The denoising is performed in the WL and FFT domain.

For the WL based processing an empirically selected Daubechies filter of order 8 (in the Matlab wavelet toolbox denoted by DB4, [11]) is used. The first 5 scales are used in the denoising process, which accesses the top 31/32 of the spectrum as band pass regions (detail) and the lower 1/32 of the spectrum as the low pass region (approximation). The band pass region as determined by the 5 scales, occupies $1/2 + 1/4 + 1/8 + 1/16 + 1/32 = 31/32$ of the spectral range leaving the remainder (1/32) as the low pass region. In the wavelet literature, band pass signals and low pass signals are denoted by detail and approximation functions, respectively.

For the FFT based processing an FFT size of 32, an overlap of 4:1 (i.e., 75 percent), and a triangular data window is used [12]. For the given data length of 512, this resulted in 61 output points for each spectral bin. Since the data is real valued, only the non-negative frequency regions are processed, with the negative spectral region being replaced with the complex conjugate of the corresponding processed positive spectral region.

6. SIMULATION RESULTS

Figures 5 through 10 show the mean squared error (MSE) performance versus SNR for the three test signals. The experiments utilize 6 different SNRs. The MSE, at each SNR, is given by

$$MSE = 1/(KN) \sum_{k=1}^K \sum_{n=1}^N (s(n) - \hat{s}(n,k))^2$$

where K is the number of realizations, N is the number of data points (i.e., fixed at 512), and $\hat{s}(n,k)$ is the k -th denoised (filtered) realizations of $s(n)$ derived from $x(n)$, the noisy data. For the generation of figure 5 through 10 and A.1 through A.6, 100 realizations, (i.e., $K = 100$), are used.

The MSE is one benchmark that can be used to establish performance. We note that when the Wiener filter (i.e., predictor) is used, then strictly interpreting the MSE results can be misleading. This is especially true as the MSE approaches a value of one. For example, it is possible for the weights of the Wiener based filter to become very small. Hence the predictor output tends to be zero. Since the power of the true signal is set to be unity, the MSE tends to be one. When a predictor is involved, it may be desirable to examine the

actual denoised output. Some representative examples (-6 and 6 dB) are given in the second part of appendix A. We note that for time delay of arrival (TDOA) estimation the MSE can be interpreted as a measure of correlation, that is as the error goes to zero, the sum of the auto correlation coefficient of the replica and of the auto correlation coefficient of the estimate tends to equal twice the cross correlation coefficient of the signal and the estimate. In this sense, it suggests that the replica (i.e., true signal) and the denoised signal correlate strongly.

Figures 5 through 10 illustrate plots of the MSE for the three test signals as a function of processing technique and SNR. All test results are obtained by pre-filtering the data with a median filter. That is prior to time-scaling or time-frequency decomposition, the data is filtered using a median filter of order 3. A 3rd order filter is the smallest possible median filter. It achieves some noise reduction without extensive distortion of transient features. The odd numbered figures show results resulting from the WL based decomposition, while the even numbered figures show results resulting from the FFT based decomposition.

The solid line (i.e., the line with circles) serves as a reference line and demonstrates the MSE performance when using only a 3rd order median filter. This median filter is the only filtering applied and is implemented in the time domain. The variance of the noise corrupting the signals is $\frac{1}{4}$, $\frac{1}{2}$, 1, 2, 4 and 10 at 6, 3, 0, -3, -6, and -10 dB, respectively. These variance values will also correspond to the MSE if no filtering is done.

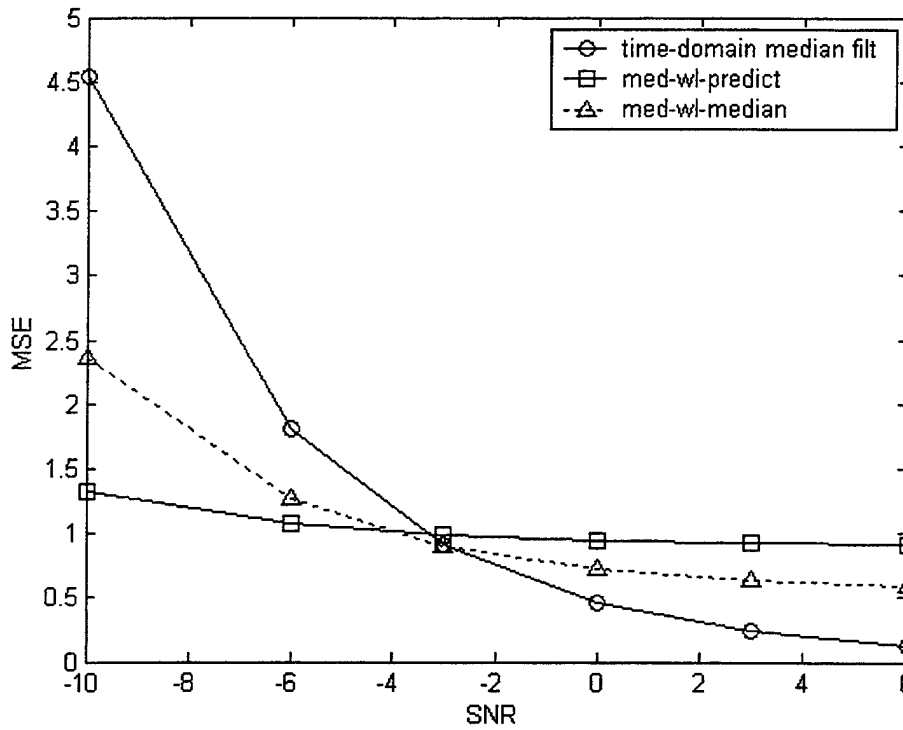


Figure 5. Signal S_1 . Pre-filtering only and pre-filtered WL decomposition.

Figure 5 and 6 show results for the test signal S_1 . Signal S_1 is the amplitude modulated sinusoid, whose amplitude increases in an RC time constant fashion. Figure 5 shows the MSE of the filtered output, using just the median filter of size 3, which reduces the variance of the data. Follow on processing via the WL based decomposition reduces the error relative to pure time domain median filtering for SNR values below -3 dB. Based on the MSE the prediction filter, applied to the wavelet coefficients, outperforms median filtering applied to the wavelet coefficients. The situation is reversed for SNR values higher than -3 dB. In this case, the predictor based scheme displays the worst performance.

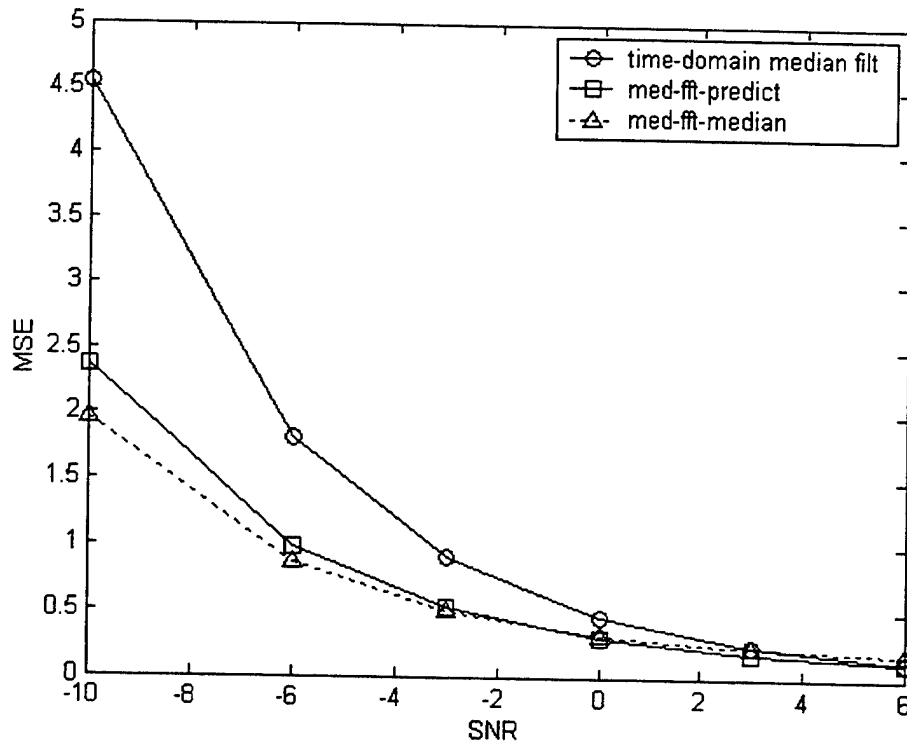


Figure 6. Signal S_1 . Pre-filtering only and pre-filtered FFT decomposition.

Figure 6 shows the performance when median pre-filtering and FFT based decomposition is used. Again the time domain median only filter output serves as a reference (i.e., solid line with circles). For all SNR levels below 6 dB, the predictor outperforms the median only filtering. For SNR levels below the 0 db level, median filtering of the FFT coefficients, in terms of the MSE, provides the best results.

Figures 7 and 8 show results for the test signal S_2 . Signal S_2 is a constant amplitude linear frequency chirped sinusoid.

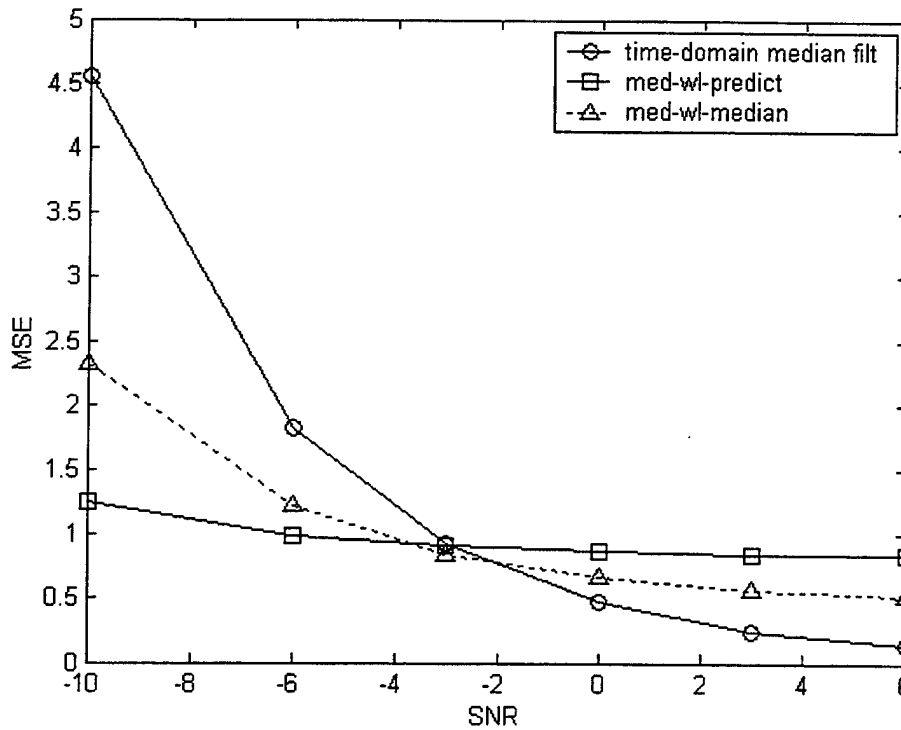


Figure 7. Signal S_2 . Pre-filtering only and pre-filtered WL decomposition.

Figure 7 shows the MSE of the filtered output, using the time domain median filter of size 3, which reduces the variance of the data. Follow on processing via the WL based decomposition reduces the error relative to just median filtering for SNR values below -3 dB. Based on the MSE, the prediction filter, when applied to wavelet coefficients, outperforms median filtering applied to wavelet coefficients for SNR values below -3 dB. The situation is reversed for SNR values higher than -3 dB. In this case, the predictor based scheme displays the worst performance. The performance is very similar to the one obtained using test signal S_1 .

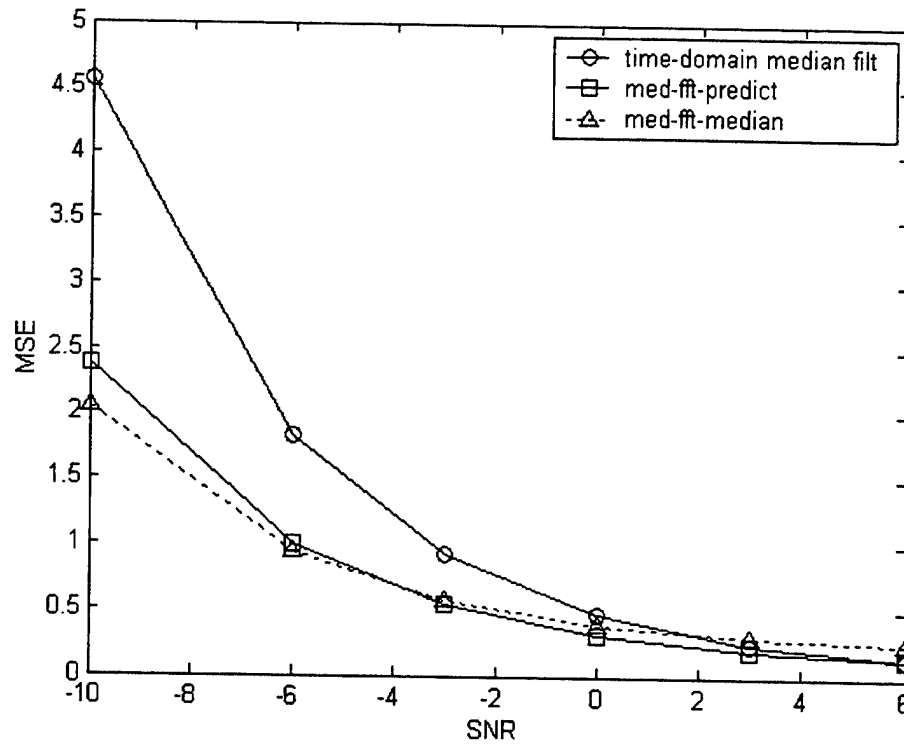


Figure 8. Signal S_2 . Pre-filtering only and pre-filtered FFT decomposition.

Figure 8 shows the performance when median pre-filtering and FFT based decomposition is used. Again the time domain median only filter output serves as a reference (i.e., solid line with circles). For all SNR levels below 6 dB, the predictor outperforms the median only filtering. For SNR levels below the -3 dB level median filtering of the FFT coefficients provides the best MSE results,.

Figures 9 and 10 show results for the test signal S_3 (i.e., the Barker coded BPSK signal).

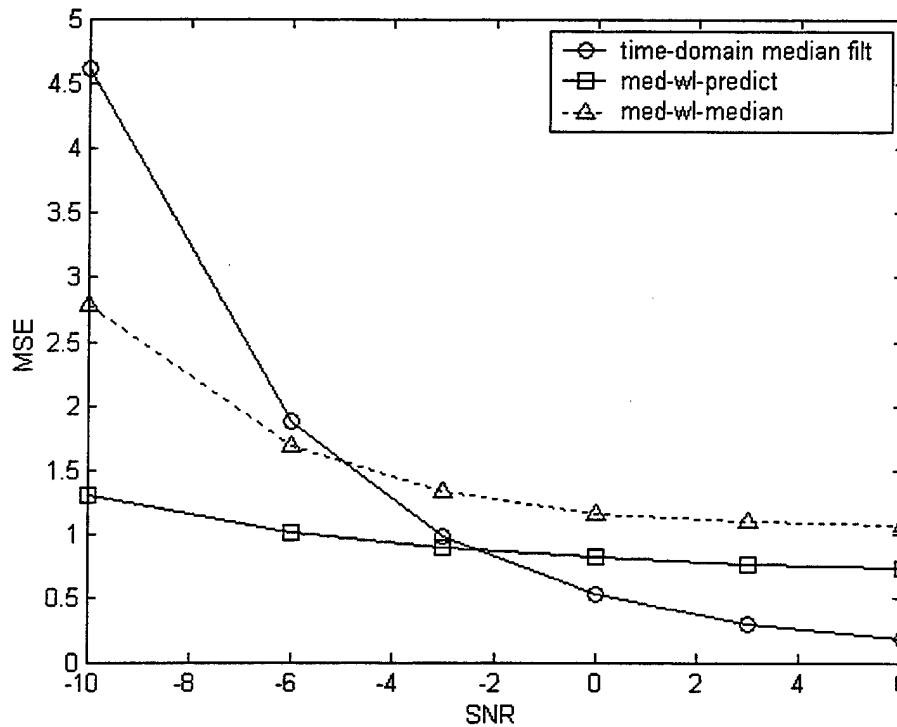


Figure 9. Figure S₃. Pre-filtering only and pre-filtered WL decomposition.

Figure 9 show that the MSE of the filtered output, using the time domain median filter of size 3, which reduces the variance of the data. For SNR values below -3 dB, follow on processing via the WL based decomposition reduces the error relative to median filtering only. Above the -3 dB level, the time domain based median filter has the edge over the WL based decomposition using either median or prediction filtering. For all SNR values under consideration, the prediction filter, as applied to the wavelet coefficients, outperforms median filtering applied to the wavelet coefficients.

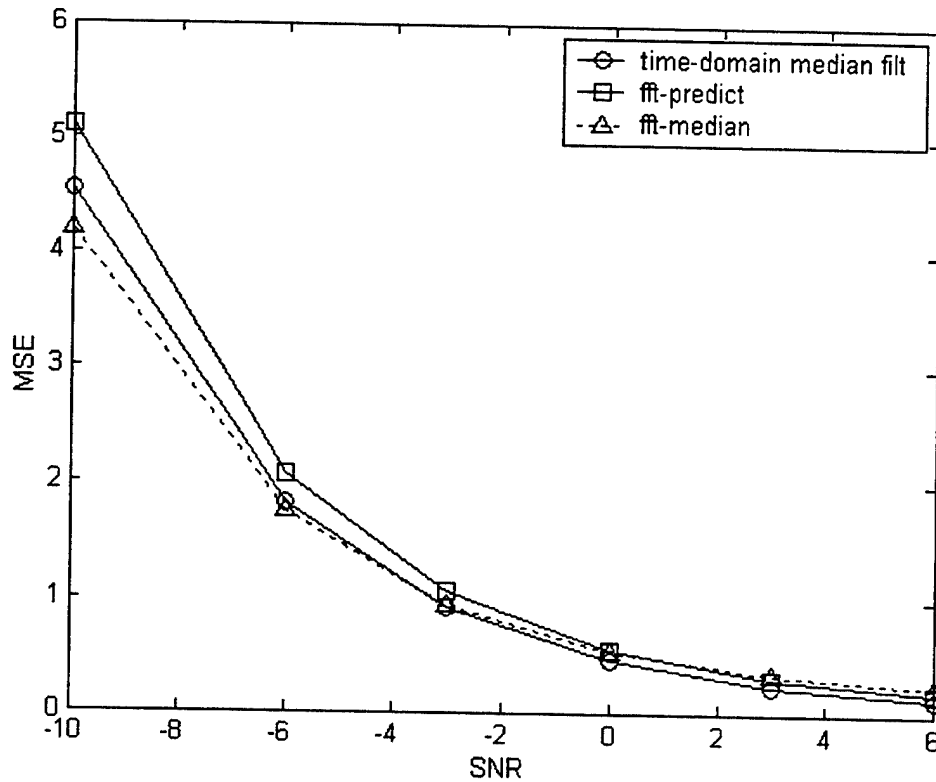


Figure 10. Signal S_3 . Pre-filtering only and pre-filtered FFT decomposition.

Figure 10 shows the performance when median pre-filtering and FFT based decomposition is used. Again the time domain median only filter output serves as a reference (i.e., solid line with circles). For all SNR levels, the predictor output has a worse MSE performance than the median only filtering. For SNR levels below the -3 dB level, in terms of the MSE, median filtering of the FFT coefficients provides the best results. The poor performance of the predictor comes as no surprise, since the phase of the sinusoid changes 180 degrees at random points in time. For SNR values larger than -3 dB, straight forward time domain median filtering achieves the best MSE results.

The data was also processed using the WL and FFT based decompositions and follow on processing without pre-filtering. The results are not as promising as the ones

when the pre-filter (i.e., time domain median filtering of order 3) is used. Plots of the MSE are provided in the first part of appendix A.

The MSE can not tell the whole story since at least in the case of a Wiener (optimal) filter the possibility exists that the filter weights become very small, hence the filter (i.e., predictor) output can become very small. This implies that the MSE will approach the power of the signal, which in the simulation is forced to be unity. It may be advisable to look at particular denoised signals to have a visual interpretation of the quality. Some typical randomly selected examples, (i.e., at -6 and 6 dB) are given in the second part of appendix A.

7. CONCLUSION

The chirped sinusoids, for SNR levels below -3 dB are best denoised using a combination of median pre-processing, FFT decomposition, and median filtering of the FFT coefficients. Above -3 dB median pre-processing, FFT decomposition and predictive filtering of the FFT coefficients have a slight edge in terms of MSE over median filtering of the FFT coefficients.

It appears that of the decomposition and processing techniques examined, time domain median filtering followed by FFT based decomposition, which in turn is followed by median filtering, provides the superior MSE performance.

The Barker coded BPSK signal is best denoised using time domain median filtering. This particular signal is very sensitive to SNR since phase reversals are more easily distorted even for SNR values greater than -3 dB.

8. REFERENCES

1. Hippenstiel, R., Haney, T., and Ha, T., "Improvement of the Time Difference of Arrival (TDOA) Estimation of GSM Signals Using Wavelets," NPS-EC-00-08, June 30, 2000, Naval Postgraduate School.
2. Hippenstiel, Ralph, *Detection Theory: Applications and Digital Signal Processing*, CRC Press, Boca Raton, FL, 2002.
3. Donahoe, D., and Johnstone, I., "Ideal Spatial Adaptation via Wavelet Shrinkage," *Biometrika*, vol. 81, pp 425-255, 1994.
4. Donahoe, D., "Denoising by Thresholding," *IEEE Information Theory*, vol. 41, pp 613-627, May 1995.
5. Donahoe, D., and Johnstone, I., "Adapting to Unknown Smoothness via Wavelet Shrinkage," *Journal of American Statistics Assoc.*, vol. 90, pp 1200-1224, December 1995.
6. Aktas, U., *Time Difference of Arrival (TDOA) Estimation Using Wavelet Based Denoising*, Master's Thesis, Naval Postgraduate School, Monterey, CA, 1999.
7. Therrien, C.W., *Discrete Random Signals and Statistical Signal Processing*. Prentice Hall, Inc., Englewood Cliffs, NJ., 1992.
8. The Matlab software, version 6.0, The Mathworks, Inc., Natick, MA, 1999.
9. WaveLab Version 802, wavelab@stat.stanford.edu
10. Andren, C., "Short PN Sequences for Direct Sequence Spread Spectrum Radios," <http://www.ss-mag.com/pdf/shortpn.pdf> , pg. 1- 4, April, 4 1997.
11. The Matlab Wavelet Toolbox, The Mathworks, Inc., Natick, MA, 1996.
12. Marple, L., Jr., *Digital Spectral Analysis with Applications*. Prentice Hall, Inc., Englewood Cliffs, NJ., 1987.

APPENDIX A

The appendix consists of two parts. The first part (figures A.1 through A.6) shows MSE results when no pre-filtering is used. That is the noisy data is FFT or WL decomposed and then processed via an optimal predictor (size 2) or a median filter (size 3). In all chirp signal test cases, i.e., figure A.1 through A.4, the results indicate that prior data manipulation (i.e., 3 point median filtering in the time domain) will out perform the schemes that do not use pre-filtering. The Barker coded BPSK, seems only to benefit from time domain median filtering only at high SNR values (i.e., 6 dB or more), see for example figure A.12. The decompositions are not very useful when denoising a signal belonging to the family characterized by S_3 (i.e., Barker coded BPSK).

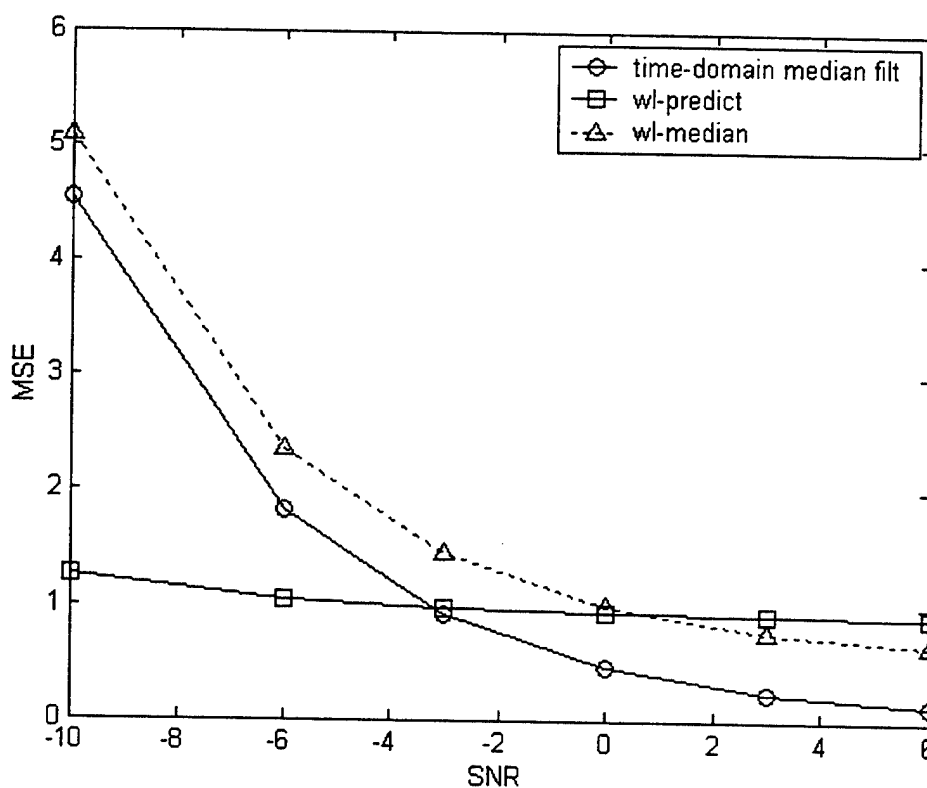


Figure A.1: No pre-processing, Signal S_1 . Time domain pre-filtering versus WL decomposition.

Figure A.1 shows results for signal S_1 . As a benchmark (solid line with circles) the time domain only median filter of order 3 is used. The other two plots show WL decomposition results that uses follow on median (order 3) and prediction filtering (size 2).

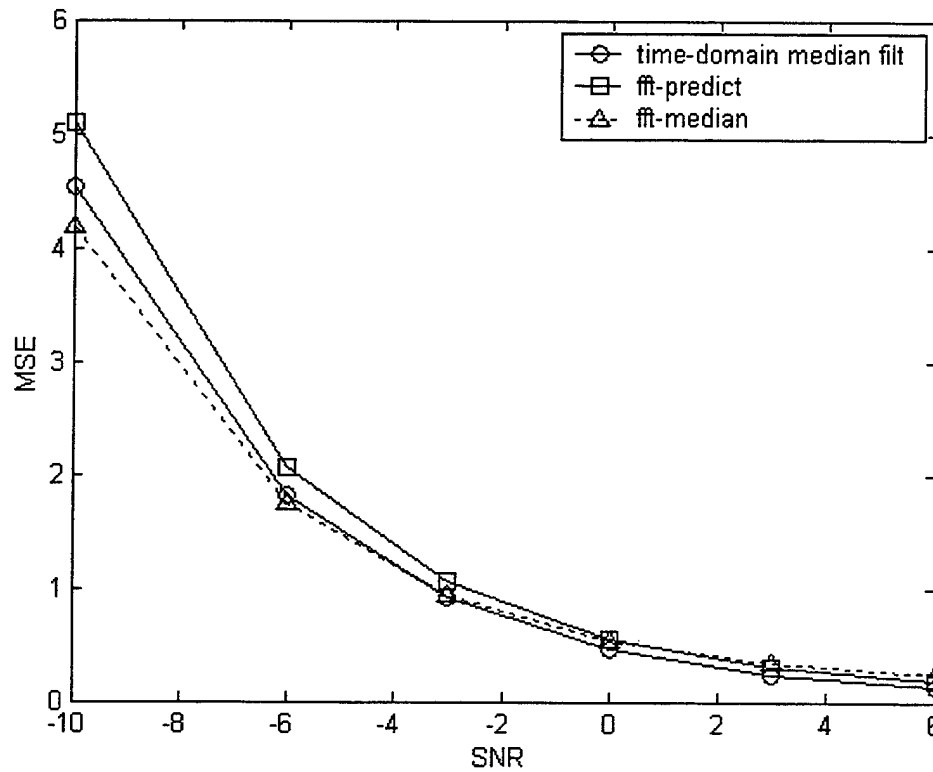


Figure A.2: No pre-processing, Signal S_1 . Time domain pre-filtering versus FFT decomposition.

Figure A.2 shows results for signal S_1 . As a benchmark (solid line with circles) the time domain only median filter of order 3 is used. The other two plots show FFT decomposition results that uses follow on median (order 3) and prediction filtering (size 2).

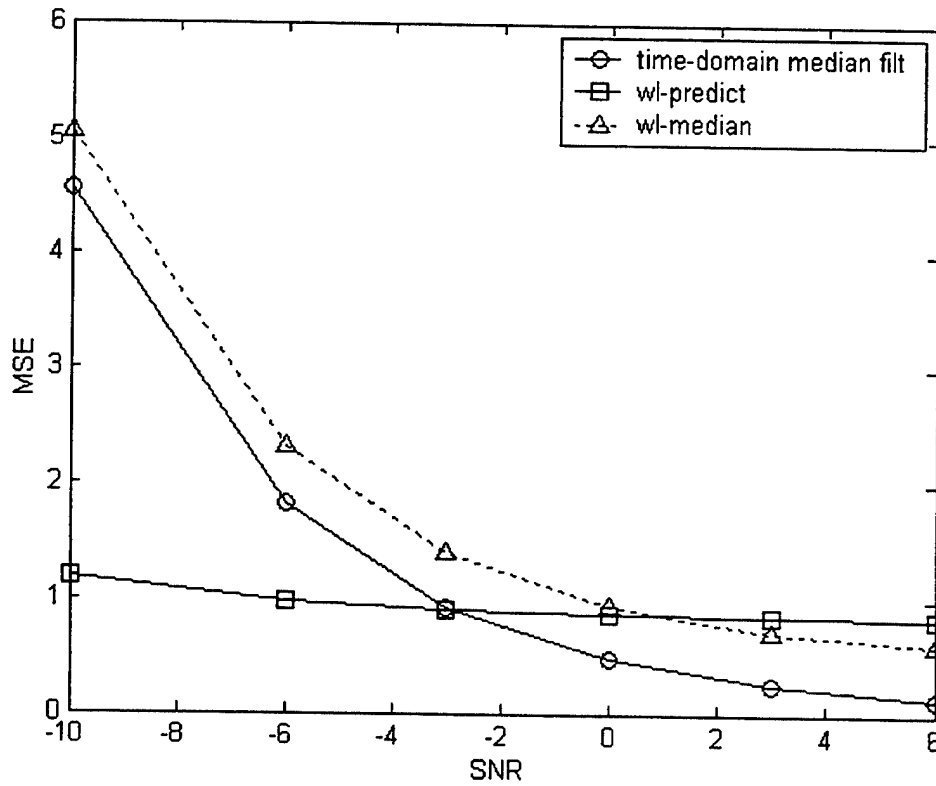


Figure A.3 : No pre-processing, Signal S_2 . Time domain pre-filtering versus WL decomposition.

Figure A.3 shows results for signal S_2 . As a benchmark (solid line with circles) the time domain only median filter of order 3 is used. The other two plots show WL decomposition results that uses follow on median (order 3) and prediction filtering (size 2).

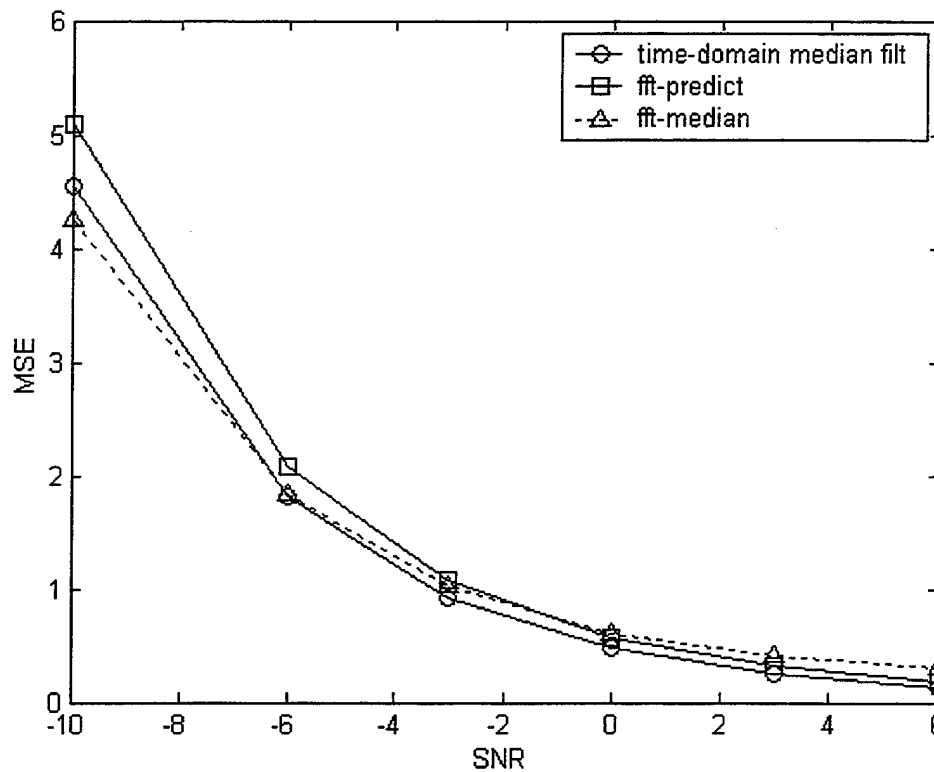


Figure A.4 : No pre-processing, Signal S_2 . Time domain pre-filtering versus FFT decomposition.

Figure A.4 shows results for signal S_2 . As a benchmark (solid line with circles) the time domain only median filter of order 3 is used. The other two plots show FFT decomposition results that uses follow on median (order 3) and prediction filtering (size 2).

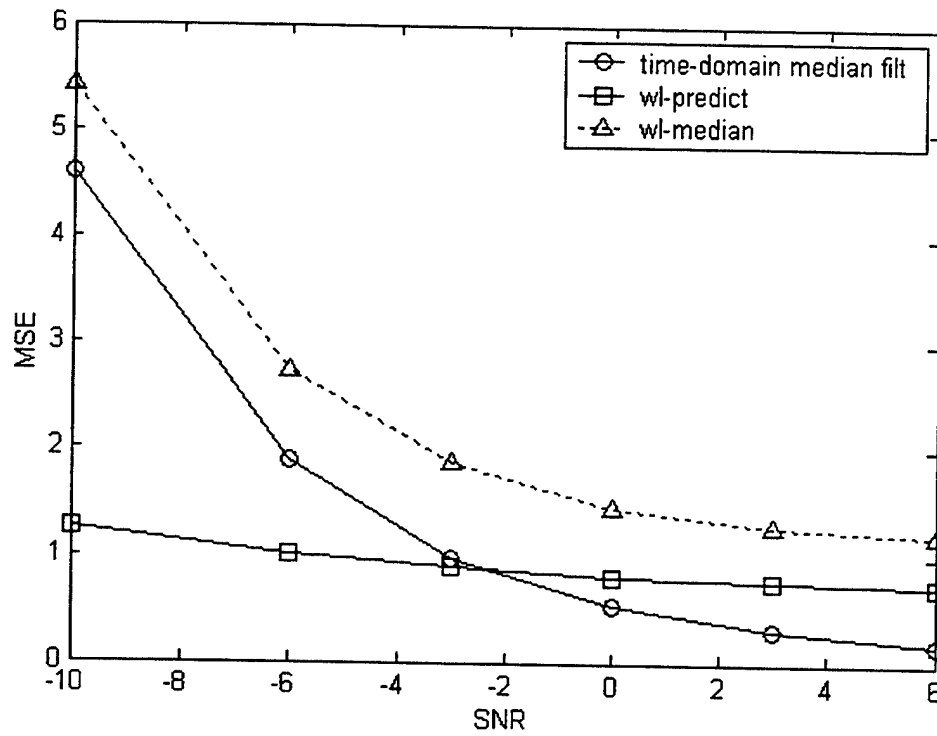


Figure A.5: No pre-processing, Signal S_3 . Time domain pre-filtering versus WL decomposition.

Figure A.5 shows results for signal S_3 . As a benchmark (solid line with circles) the time domain only median filter of order 3 is used. The other two plots show WL decomposition results that uses follow on median (order 3) and prediction filtering (size 2).

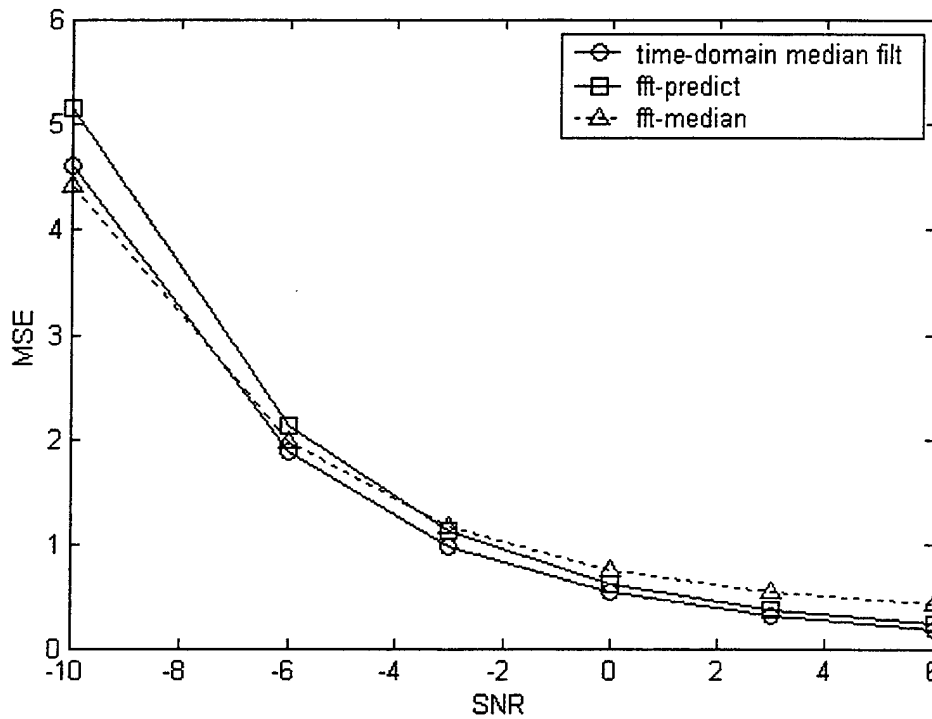


Figure A.6: No pre-processing, Signal S_3 . Time domain pre-filtering versus FFT decomposition.

Figure A.6 shows results for signal S_3 . As a benchmark (solid line with circles) the time domain only median filter of order 3 is used. The other two plots show FFT decomposition results that uses follow on median (order 3) and prediction filtering (size 2).

The second part of appendix A serves as an illustration as to how the signals (original, noisy signal, denoised signal) look like in the time domain for a few selected values of SNR. The SNR values selected are -6 and 6 dB. Since there are many realizations at each SNR, only the first realization is used in the plots (i.e., a random

member of the ensemble). These plots can provide some insight into time domain performance since the MSE results can be misleading when it comes to the optimal predictor implementation.

Figures A.7 through A.9 have an SNR of -6 dB and represent S1, S2, and S3, respectively. Figure A.10 through A.12 have an SNR of 6 dB and represent signals S1, S2, and S3, respectively. Each plot consists of 7 subplots. The subplots, for the purpose of this discussion, are referred to in the same sense as the members of a matrix, that is subplot (row, column). Subplot (1,1) and (1,2) show the signal and noisy signal, respectively. Subplot (2,1) shows the time domain median filtered result. Subplot (3,1) and (4,1) show results when the data is pre-processed and wavelet decomposed. Subplot (3,1) uses prediction on the wavelet coefficients, while (4,1) uses median filtering on the wavelet coefficients. Subplot (3,2) and (4,2) uses pre-processing and FFT decomposition. Subplot (3,2) uses prediction on the FFT coefficients, while (4,1) uses median filtering on the FFT coefficients.

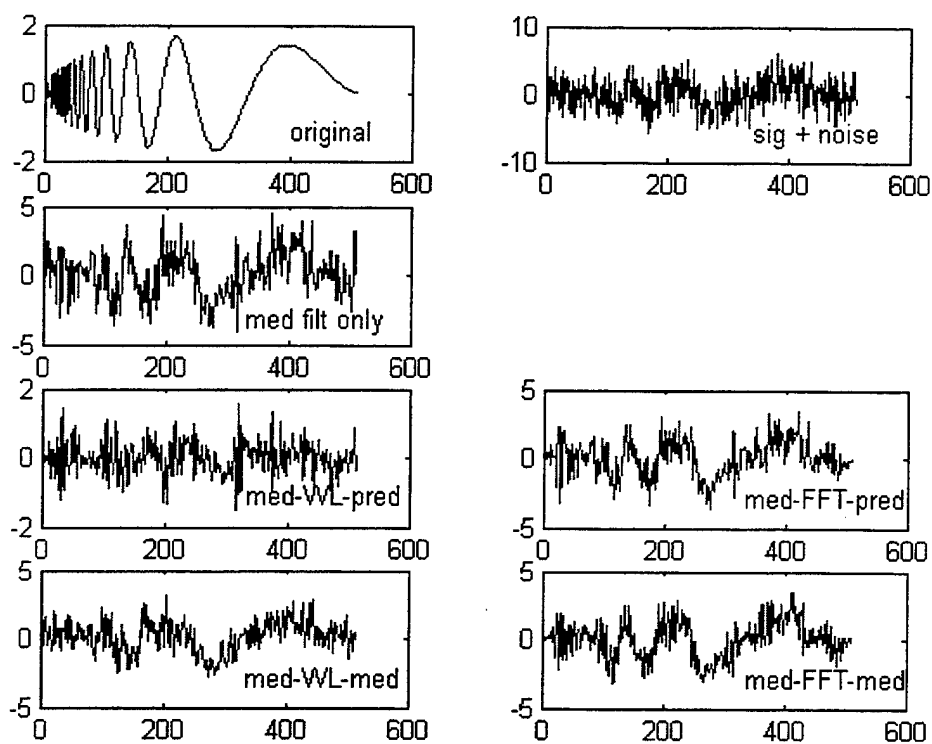


Figure A.7: Signal S_1 at -6 dB

Figure A.7 shows a typical time domain representation of signal S_1 , at an SNR value of -6 dB. The signal is pre-processed (i.e., median filtering of order 3 is applied in the time domain).

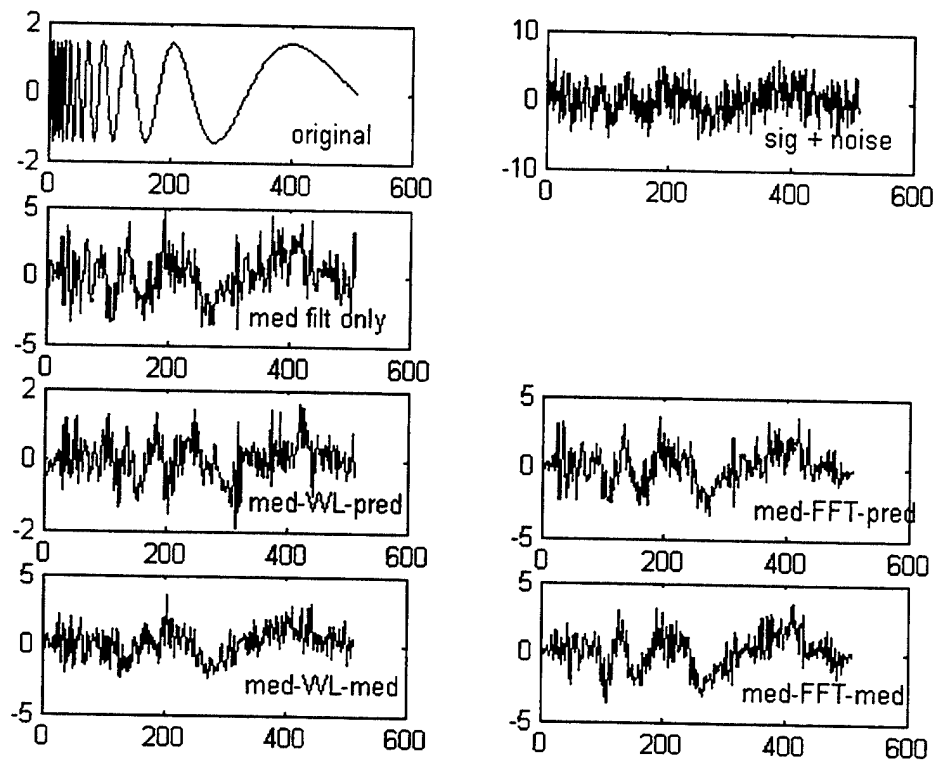


Figure A.8: Signal S_2 at -6 dB

Figure A.8 shows a typical time domain representation of signal S_2 , at an SNR value of -6 dB. The signal is pre-processed (i.e., median filtering of order 3 is applied in the time domain).

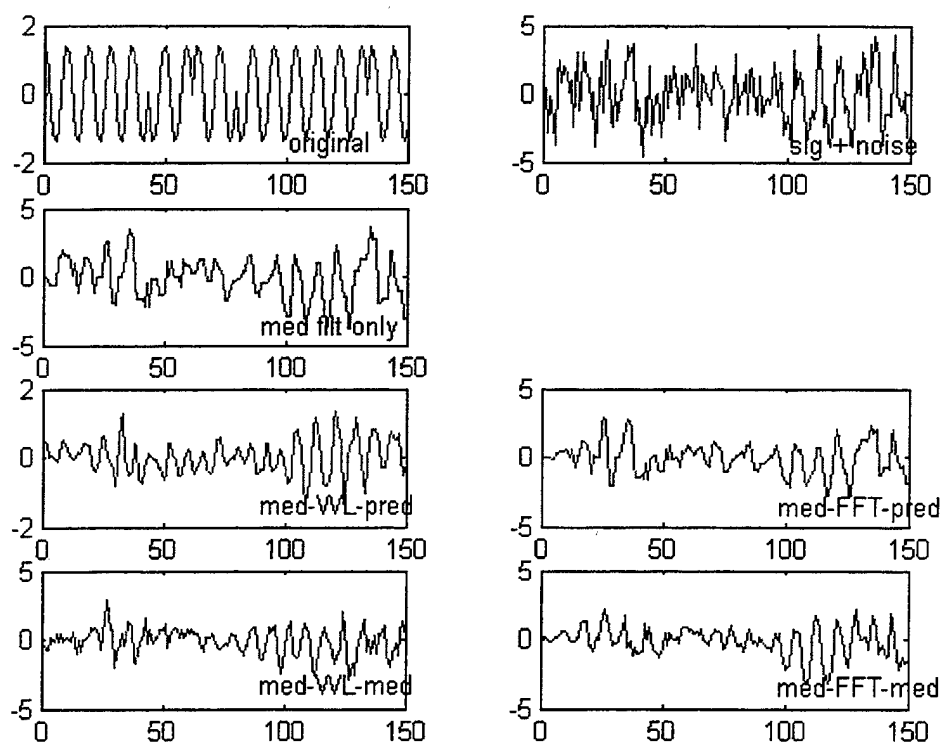


Figure A.9: Signal S_3 at -6dB (first 150 samples)

Figure A.9 shows a typical time domain representation of signal S_3 , at an SNR value of -6 dB . Only the first 150 data points are used to show the phase transition points more clearly. The signal is pre-processed (i.e., median filtering of order 3 is applied in the time domain).

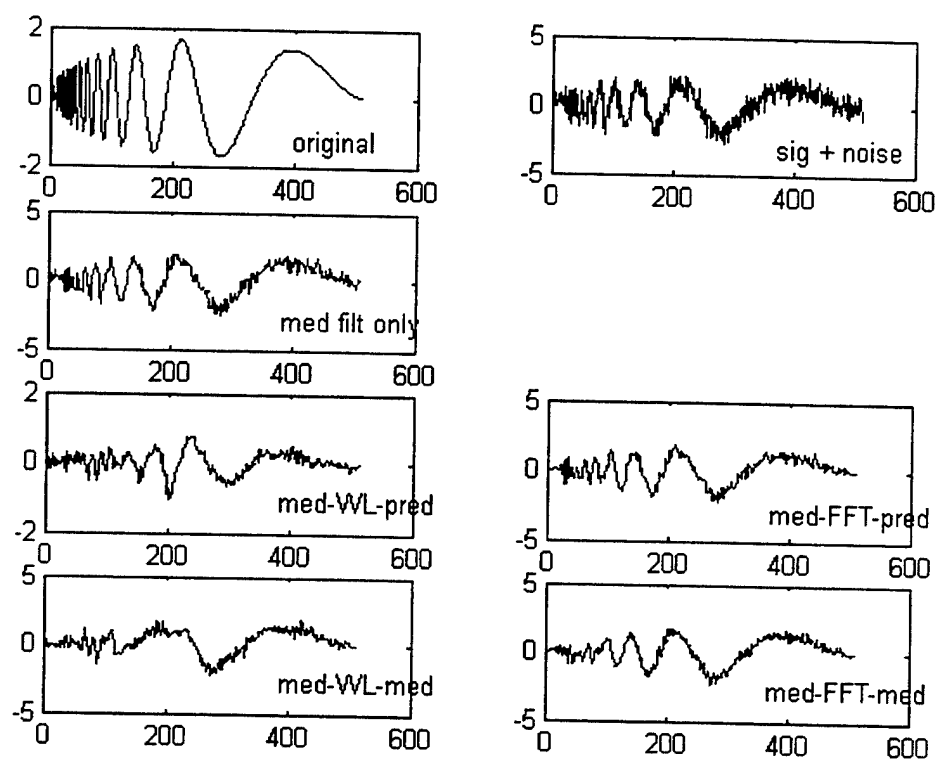


Figure A.10: Signal S_1 at 6 dB

Figure A.10 shows a typical time domain representation of signal S_1 , at an SNR value of 6 dB. The signal is pre-processed (i.e., median filtering of order 3 is applied in the time domain).

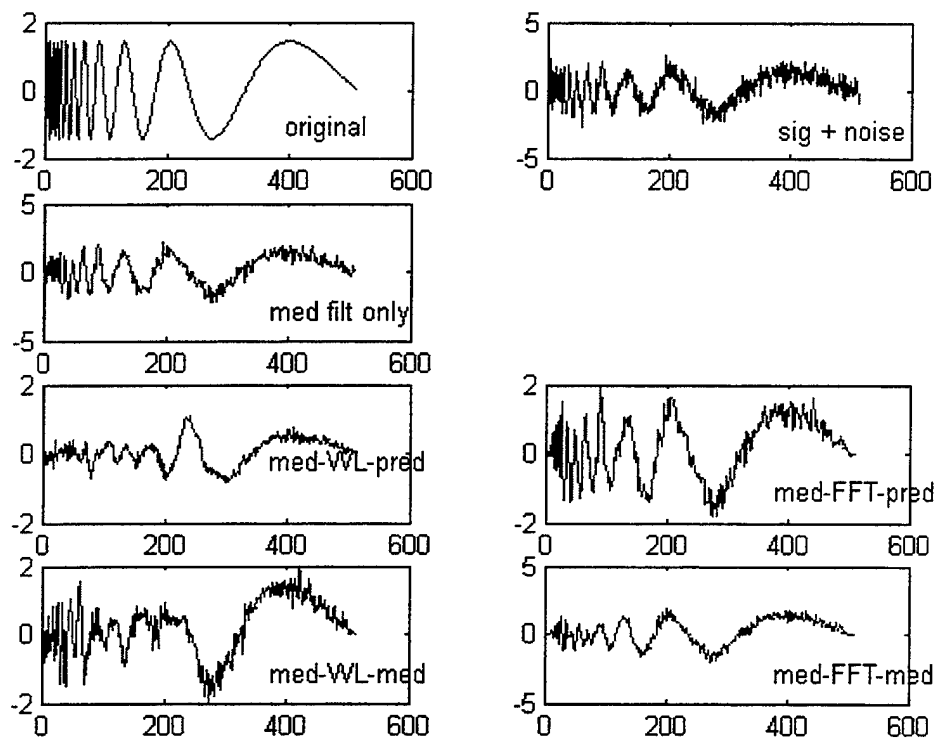


Figure A.11: Signal S_2 at 6 dB

Figure A.11 shows a typical time domain representation of signal S_2 , at an SNR value of 6 dB. The signal is pre-processed (i.e., median filtering of order 3 is applied in the time domain).

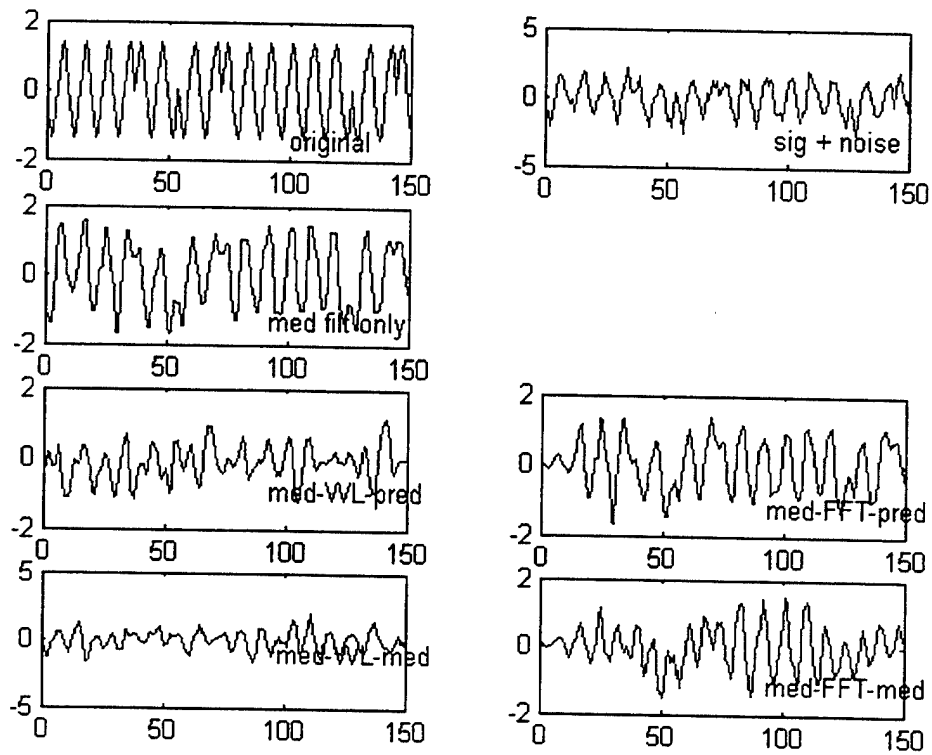


Figure A.12: Signal S_3 at 6 dB (first 150 samples)

Figure A.12 shows a typical time domain representation of signal S_3 , at an SNR value of 6 dB. Only the first 150 data points are used to show the phase transition points more clearly. The signal is pre-processed (i.e., median filtering of order 3 is applied in the time domain).

APPENDIX B

Matlab code:

nois_rem_2.m

```
%%%%%%%%%%
% program nois_rem_2
%%%%%%%%%%
% creates and denoises 3 test signals for a set of SNR's.
% Doppler signal+ chirp with RC time constant envelope
% approaching a constant, a constant envelope chirp, and
% a Barker coded BPSK signal. All signals have unit power.
% calls makesignal,denoisewl, predi,dnswlmed
% written by Ralph Hippenstiel
% wavelets used are Db4 (Daubechies order 8)
% FFTs use a triangular window and 4:1 overlap (75% overlap)
%%%%%%%%%%

clear
nx=input('enter number of scales nx = ');
p=input('enter predictor size = ');
nd=512;
nr=input('enter number of realizations (i.e., 100) = ');

%create the signal s1
s1=makesignal('Doppler',nd);
z1=length(s1);% z1 is length of the data = nd
pwr1=1/z1*sum(s1.^2);
s1=s1/sqrt(pwr1);%normalized power

%create the signal s2
tim=[0.01:0.01:6.12];
xchirp=sin(34.7./tim);
```

```

s2=xchirp(41:41-1+z1);
pwr2=1/z1*sum(s2.^2);
s2=s2/sqrt(pwr2);%normalized power

%define SNR
sn=[-10,-6,-3,0,3,6];
SNR=10.^(sn./10);

err_1=[];
err_2=[];
err_3=[];

err_wl_pred1=[];
err_wl_pred2=[];
err_wl_pred3=[];
er_wl_pred1=[];
er_wl_pred2=[];
er_wl_pred3=[];

err_wl_med1=[];
err_wl_med2=[];
err_wl_med3=[];
er_wl_med1=[];
er_wl_med2=[];
er_wl_med3=[];

err_fft_pred1=[];
err_fft_pred2=[];
err_fft_pred3=[];
er_fft_pred1=[];
er_fft_pred2=[];
er_fft_pred3=[];

err_fft_med1=[];
err_fft_med2=[];
err_fft_med3=[];
er_fft_med1=[];
er_fft_med2=[];
er_fft_med3=[];

for i=1:nr; %nr = no. of realizations

    %create Gaussian noise, zero mean & unit variance
    n=randn(1,z1);

```

```

%create the signal s3
%w=[ones(18,1); -ones(18,1); ones(6,1); -ones(12,1); ones(6,1); -ones(6,1)];
w=[ones(3*18,1); -ones(18,1); ones(18,1)];
%bi=sign(rand(1,8)-0.5);
%vi=[bi(1)*w;bi(2)*w;bi(3)*w;bi(4)*w;bi(5)*w;bi(6)*w;bi(7)*w;bi(8)*w;];
%si=sin(2*pi/6*[0:527]);
bi=sign(rand(1,6)-0.5);
vi=[bi(1)*w;bi(2)*w;bi(3)*w;bi(4)*w;bi(5)*w;bi(6)*w]';
si=sin(2*pi/9*[0:539]);
s3=si.*vi;
%tsi=rand(1,1)*16;
tsi=rand(1,1)*27;
ltsi=ceil(tsi);
s3=s3(ltsi:ltsi+511);
pwr3=1/z1*sum(s3.^2);
s3=s3/sqrt(pwr3);% normalized power

```

```

for k=1:length(SNR); %no. of SNR's
    %scale noise for req. SNR
    ora=SNR(k);
    noi=sqrt(1/(ora))*n;
    npwr=1/z1*sum(noi.^2);

```

```

%xi, i=1,2,3 is the raw data
x1=s1+noi;
x2=s2+noi;
x3=s3+noi;

```

```

%xxi, i=1,2,3 is median filtered (size 3) data
% pre-filter
xx1=medfilt1(x1,3);
xx2=medfilt1(x2,3);
xx3=medfilt1(x3,3);

```

```

%ONLY pre-filtered error results
err_1(k,i)=1/z1*sum((s1-xx1).^2);
err_2(k,i)=1/z1*sum((s2-xx2).^2);
err_3(k,i)=1/z1*sum((s3-xx3).^2);

```

```

% wavelet processing with prediction
% pre-filtered:
x1_real=denoisewl(xx1,p,nx);
err_wl_pred1(k,i)=1/z1*sum((s1-x1_real).^2);

```



```

x2_real=denoisewl(xx2,p,nx);
err_wl_pred2(k,i)=1/z1*sum((s2-x2_real).^2);
x3_real=denoisewl(xx3,p,nx);
err_wl_pred3(k,i)=1/z1*sum((s3-x3_real).^2);

```

 % No pre-filter:

```

x1_real=denoisewl(x1,p,nx);
er_wl_pred1(k,i)=1/z1*sum((s1-x1_real).^2);
x2_real=denoisewl(x2,p,nx);
er_wl_pred2(k,i)=1/z1*sum((s2-x2_real).^2);
x3_real=denoisewl(x3,p,nx);
er_wl_pred3(k,i)=1/z1*sum((s3-x3_real).^2);

```

% wavelet processing with median filtering

 % pre-filtered:

```

x1_real=dnswlmed(xx1,nx);
err_wl_med1(k,i)=1/z1*sum((s1-x1_real).^2);
x2_real=dnswlmed(xx2,nx);
err_wl_med2(k,i)=1/z1*sum((s2-x2_real).^2);
x3_real=dnswlmed(xx3,nx);
err_wl_med3(k,i)=1/z1*sum((s3-x3_real).^2);

```

 % No pre-filter:

```

x1_real=dnswlmed(x1,nx);
er_wl_med1(k,i)=1/z1*sum((s1-x1_real).^2);
x2_real=dnswlmed(x2,nx);
er_wl_med2(k,i)=1/z1*sum((s2-x2_real).^2);
x3_real=dnswlmed(x3,nx);
er_wl_med3(k,i)=1/z1*sum((s3-x3_real).^2);

```

% FFT processing with prediction

 % pre-filtered:

```

x1_real=predi(xx1,p);
err_fft_pred1(k,i)=1/z1*sum((s1-x1_real).^2);
x2_real=predi(xx2,p);
err_fft_pred2(k,i)=1/z1*sum((s2-x2_real).^2);
x3_real=predi(xx3,p);
err_fft_pred3(k,i)=1/z1*sum((s3-x3_real).^2);

```

 %No Prefilter:

```

x1_real=predi(x1,p);
er_fft_pred1(k,i)=1/z1*sum((s1-x1_real).^2);
x2_real=predi(x2,p);

```

```

er_fft_pred2(k,i)=1/z1*sum((s2-x2_real).^2);
x3_real=predi(x3,p);
er_fft_pred3(k,i)=1/z1*sum((s3-x3_real).^2);

```

```

%FFT processing with median filtering

```

```

    % pre-filtered:

```

```

x1_real=denoise_med(xx1);
err_fft_med1(k,i)=1/z1*sum((s1-x1_real).^2);
x2_real=denoise_med(xx2);
err_fft_med2(k,i)=1/z1*sum((s2-x2_real).^2);
x3_real=denoise_med(xx3);
err_fft_med3(k,i)=1/z1*sum((s3-x3_real).^2);

```

```

    % No pre-filter:

```

```

x1_real=denoise_med(x1);
er_fft_med1(k,i)=1/z1*sum((s1-x1_real).^2);
x2_real=denoise_med(x2);
er_fft_med2(k,i)=1/z1*sum((s2-x2_real).^2);
x3_real=denoise_med(x3);
er_fft_med3(k,i)=1/z1*sum((s3-x3_real).^2);
end
end

```

```

% compute statistical averages

```

```

me1=mean(err_1');
me2=mean(err_2');
me3=mean(err_3');

```

```

me_wl_pred1=mean(err_wl_pred1');
me_wl_pred2=mean(err_wl_pred2');
me_wl_pred3=mean(err_wl_pred3');

```

```

e_wl_pred1=mean(er_wl_pred1');
e_wl_pred2=mean(er_wl_pred2');
e_wl_pred3=mean(er_wl_pred3');

```

```

me_wl_med1=mean(err_wl_med1');
me_wl_med2=mean(err_wl_med2');
me_wl_med3=mean(err_wl_med3');

```

```

e_wl_med1=mean(er_wl_med1');
e_wl_med2=mean(er_wl_med2');
e_wl_med3=mean(er_wl_med3');

```

```

me_fft_pred1=mean(err_fft_pred1');
me_fft_pred2=mean(err_fft_pred2');
me_fft_pred3=mean(err_fft_pred3');

```

```
e_fft_pred1=mean(er_fft_pred1');  
e_fft_pred2=mean(er_fft_pred2');  
e_fft_pred3=mean(er_fft_pred3');
```

```
me_fft_med1=mean(err_fft_med1');  
me_fft_med2=mean(err_fft_med2');  
me_fft_med3=mean(err_fft_med3');
```

```
e_fft_med1=mean(er_fft_med1');  
e_fft_med2=mean(er_fft_med2');  
e_fft_med3=mean(er_fft_med3');
```

Makesignal.m

```
function sig = MakeSignal(Name,n)
% MakeSignal -- Make artificial signal
% Usage
% sig = MakeSignal(Name,n)
% Inputs
% Name string: 'HeaviSine', 'Bumps', 'Blocks',
%             'Doppler', 'Ramp', 'Cusp', 'Sing', 'HiSine',
%             'LoSine', 'LinChirp', 'TwoChirp', 'QuadChirp',
%             'MishMash', 'WernerSorrows' (Heisenberg),
%             'Leopold' (Kronecker), 'Piece-Regular' (Piece-Wise Smooth),
%             'Riemann', 'HypChirps', 'LinChirps', 'Chirps', 'Gabor'
%             'sineoneoverx', 'Cusp2', 'SmoothCusp', 'Gaussian'
%             'Piece-Polynomial' (Piece-Wise 3rd degree polynomial)
% n desired signal length
% Outputs
% sig 1-d signal
%
% References
% Various articles of D.L. Donoho and I.M. Johnstone
%
if nargin > 1,
    t = (1:n) ./n;
end
if strcmp(Name,'HeaviSine'),
    sig = 4.*sin(4*pi.*t);
    sig = sig - sign(t - .3) - sign(.72 - t);
elseif strcmp(Name,'Bumps'),
    pos = [ .1 .13 .15 .23 .25 .40 .44 .65 .76 .78 .81];
    hgt = [ 4 5 3 4 5 4.2 2.1 4.3 3.1 5.1 4.2];
    wth = [.005 .005 .006 .01 .01 .03 .01 .01 .005 .008 .005];
    sig = zeros(size(t));
    for j = 1:length(pos)
        sig = sig + hgt(j)./( 1 + abs((t - pos(j))./wth(j))).^4;
    end
elseif strcmp(Name,'Blocks'),
    pos = [ .1 .13 .15 .23 .25 .40 .44 .65 .76 .78 .81];
    hgt = [4 (-5) 3 (-4) 5 (-4.2) 2.1 4.3 (-3.1) 2.1 (-4.2)];
    sig = zeros(size(t));
    for j = 1:length(pos)
        sig = sig + (1 + sign(t-pos(j))).*(hgt(j)/2) ;
    end
elseif strcmp(Name,'Doppler'),
    sig = sqrt(t.*(1-t)).*sin((2*pi*1.05) ./ (t+.05));
elseif strcmp(Name,'Ramp'),
```

```

    sig = t - (t >= .37);
elseif strcmp(Name,'Cusp'),
    sig = sqrt(abs(t - .37));
elseif strcmp(Name,'Sing'),
    k = floor(n * .37);
    sig = 1 ./abs(t - (k+.5)/n);
elseif strcmp(Name,'HiSine'),
    sig = sin( pi * (n * .6902) .* t);
elseif strcmp(Name,'LoSine'),
    sig = sin( pi * (n * .3333) .* t);
elseif strcmp(Name,'LinChirp'),
    sig = sin(pi .* t .* ((n * .500) .* t));
elseif strcmp(Name,'TwoChirp'),
    sig = sin(pi .* t .* (n .* t)) + sin((pi/3) .* t .* (n .* t));
elseif strcmp(Name,'QuadChirp'),
    sig = sin( (pi/3) .* t .* (n .* t.^2));
elseif strcmp(Name,'MishMash'), % QuadChirp + LinChirp + HiSine
    sig = sin( (pi/3) .* t .* (n .* t.^2)) ;
    sig = sig + sin( pi * (n * .6902) .* t);
    sig = sig + sin(pi .* t .* (n * .125 .* t));
elseif strcmp(Name,'WernerSorrows'),
    sig = sin( pi .* t .* (n/2 .* t.^2)) ;
    sig = sig + sin( pi * (n * .6902) .* t);
    sig = sig + sin(pi .* t .* (n .* t));
    pos = [ .1 .13 .15 .23 .25 .40 .44 .65 .76 .78 .81];
    hgt = [ 4 5 3 4 5 4.2 2.1 4.3 3.1 5.1 4.2];
    wth = [.005 .005 .006 .01 .01 .03 .01 .01 .005 .008 .005];
    for j =1:length(pos)
        sig = sig + hgt(j)./( 1 + abs((t - pos(j))./wth(j))).^4;
    end
elseif strcmp(Name,'Leopold'),
    sig = (t == floor(.37 * n)/n); % Kronecker
elseif strcmp(Name,'Riemann'),
    sqn = round(sqrt(n));
    sig = t .* 0; % Riemann's Non-differentiable Function
    sig((1:sqn).^2) = 1. ./ (1:sqn);
    sig = real(ifft(sig));
elseif strcmp(Name,'HypChirps'), % Hyperbolic Chirps of Mallat's book
    alpha = 15*n*pi/1024;
    beta = 5*n*pi/1024;
    t = (1.001:1:n+.001)./n;
    f1 = zeros(1,n);
    f2 = zeros(1,n);
    f1 = sin(alpha./(8-t)).*(0.1<t).*(t<0.68);
    f2 = sin(beta./(8-t)).*(0.1<t).*(t<0.75);
    M = round(0.65*n);

```

```

P      = floor(M/4);
enveloppe = ones(1,M); % the rising cutoff function
enveloppe(1:P) = (1+sin(-pi/2+((1:P)-ones(1,P))./(P-1)*pi))/2;
enveloppe(M-P+1:M) = reverse(enveloppe(1:P));
env      = zeros(1,n);
env(ceil(n/10):M+ceil(n/10)-1) = enveloppe(1:M);
sig      = (f1+f2).*env;
elseif strcmp(Name,'LinChirps'), % Linear Chirps of Mallat's book
b        = 100*n*pi/1024;
a        = 250*n*pi/1024;
t        = (1:n)/n;
A1       = sqrt((t-1/n).*(1-t));
sig      = A1.*(cos((a*(t).^2)) + cos((b*t+a*(t).^2)));
elseif strcmp(Name,'Chirps'), % Mixture of Chirps of Mallat's book
t        = (1:n)/n.*10.*pi;
f1       = cos(t.^2*n/1024);
a        = 30*n/1024;
t        = (1:n)/n.*pi;
f2       = cos(a.*(t.^3));
f2       = reverse(f2);
ix       = (-n:n)/n.*20;
g        = exp(-ix.^2*4*n/1024);
i1       = (n/2+1:n/2+n);
i2       = (n/8+1:n/8+n);
j        = (1:n)/n;
f3       = g(i1).*cos(50.*pi.*j*n/1024);
f4       = g(i2).*cos(350.*pi.*j*n/1024);
sig      = f1+f2+f3+f4;
enveloppe = ones(1,n); % the rising cutoff function
enveloppe(1:n/8) = (1+sin(-pi/2+((1:n/8)-ones(1,n/8))./(n/8-1)*pi))/2;
enveloppe(7*n/8+1:n) = reverse(enveloppe(1:n/8));
sig      = sig.*enveloppe;
elseif strcmp(Name,'Gabor'), % two modulated Gabor functions in
                                % Mallat's book
N = 512;
t = (-N:N)*5/N;
j = (1:N)/N;
g = exp(-t.^2*20);
i1 = (2*N/4+1:2*N/4+N);
i2 = (N/4+1:N/4+N);
sig1 = 3*g(i1).*exp(i*N/16.*pi.*j);
sig2 = 3*g(i2).*exp(i*N/4.*pi.*j);
sig = sig1+sig2;
elseif strcmp(Name,'sineoneoverx'), % sin(1/x) in Mallat's book
N = 1024;
i = (-N+1:N);

```

```

        i(N) = 1/100;
        i = i./(N-1);
        sig = sin(1.5./(i));
        sig = sig(513:1536);
elseif strcmp(Name,'Cusp2'),
    N = 64;
    i = (1:N)/N;
    x = (1-sqrt(i)) + i/2 -.5;
    M = 8*N;
    sig = zeros(1,M);
    sig(M-1.5.*N+1:M-.5*N) = x;
    sig(M-2.5*N+2:M-1.5.*N+1) = reverse(x);
    sig(3*N+1:3*N + N) = .5*ones(1,N);
elseif strcmp(Name,'SmoothCusp'),
    sig = MakeSignal('Cusp2');
    N = 64;
    M = 8*N;
    t = (1:M)/M;
    sigma = 0.01;
    g = exp(-.5.*(abs(t-.5)/sigma).^2)/sigma./sqrt(2*pi);
    g = fftshift(g);
    sig2 = iconv(g',sig)/M;
elseif strcmp(Name,'Piece-Regular'),
    sig1=-15*MakeSignal('Bumps',n);
    t = (1:fix(n/12)) ./fix(n/12);
    sig2=-exp(4*t);
    t = (1:fix(n/7)) ./fix(n/7);
    sig5=exp(4*t)-exp(4);
    t = (1:fix(n/3)) ./fix(n/3);
    sigma=6/40;
    sig6=-70*exp(-((t-1/2).*(t-1/2))/(2*sigma^2));
    sig(1:fix(n/7))= sig6(1:fix(n/7));
    sig((fix(n/7)+1):fix(n/5))=0.5*sig6((fix(n/7)+1):fix(n/5));
    sig((fix(n/5)+1):fix(n/3))=sig6((fix(n/5)+1):fix(n/3));
    sig((fix(n/3)+1):fix(n/2))=sig1((fix(n/3)+1):fix(n/2));
    sig((fix(n/2)+1):(fix(n/2)+fix(n/12)))=sig2;
    sig((fix(n/2)+2*fix(n/12)):-1:(fix(n/2)+fix(n/12)+1))=sig2;
    sig(fix(n/2)+2*fix(n/12)+fix(n/20)+1:(fix(n/2)+2*fix(n/12)+3*fix(n/20)))=...
    -ones(1,fix(n/2)+2*fix(n/12)+3*fix(n/20)-fix(n/2)-2*fix(n/12)-fix(n/20))*25;
    k=fix(n/2)+2*fix(n/12)+3*fix(n/20);
    sig((k+1):(k+fix(n/7)))=sig5;
    diff=n-5*fix(n/5);
    sig(5*fix(n/5)+1:n)=sig(diff:-1:1);
    % zero-mean
    bias=sum(sig)/n;
    sig=bias-sig;

```

```

elseif strcmp(Name,'Piece-Polynomial'),
    t = (1:fix(n/5)) ./fix(n/5);
    sig1=20*(t.^3+t.^2+4);
    sig3=40*(2.*t.^3+t) + 100;
    sig2=10.*t.^3 + 45;
    sig4=16*t.^2+8.*t+16;
    sig5=20*(t+4);
    sig6(1:fix(n/10))=ones(1,fix(n/10));
    sig6=sig6*20;
    sig(1:fix(n/5))=sig1;
    sig(2*fix(n/5):-1:(fix(n/5)+1))=sig2;
    sig((2*fix(n/5)+1):3*fix(n/5))=sig3;
    sig((3*fix(n/5)+1):4*fix(n/5))=sig4;
    sig((4*fix(n/5)+1):5*fix(n/5))=sig5(fix(n/5):-1:1);
    diff=n-5*fix(n/5);
    sig(5*fix(n/5)+1:n)=sig(diff:-1:1);
    %sig((fix(n/20)+1):(fix(n/20)+fix(n/10)))=-ones(1,fix(n/10))*20;
    sig((fix(n/20)+1):(fix(n/20)+fix(n/10)))=ones(1,fix(n/10))*10;
    sig((n-fix(n/10)+1):(n+fix(n/20)-
fix(n/10)))=ones(1,fix(n/20))*150;
    % zero-mean
    bias=sum(sig)/n;
    sig=sig-bias;
elseif strcmp(Name,'Gaussian'),
    sig=GWN(n,beta);
    g=zeros(1,n);
    lim=alpha*n;
    mult=pi/(2*alpha*n);
    g(1:lim)=(cos(mult*(1:lim))).^2;
    g((n/2+1):n)=g((n/2):-1:1);
    g = rmshift(g,n/2);
    g=g/norm(g);
    sig=iconv(g,sig);
else
    disp(sprintf('MakeSignal: I don*t recognize <<%s>>',Name))
    disp('Allowable Names are:')
    disp('HeaviSine'),
    disp('Bumps'),
    disp('Blocks'),
    disp('Doppler'),
    disp('Ramp'),
    disp('Cusp'),
    disp('Crease'),
    disp('Sing'),
    disp('HiSine'),
    disp('LoSine'),

```



```

disp('LinChirp'),
disp('TwoChirp'),
disp('QuadChirp'),
disp('MishMash'),
disp('WernerSorrows'),
disp('Leopold'),
disp('Sing'),
disp('HiSine'),
disp('LoSine'),
disp('LinChirp'),
disp('TwoChirp'),
disp('QuadChirp'),
disp('MishMash'),
disp('WernerSorrows'),
disp('Leopold'),
disp('Riemann'),
disp('HypChirps'),
disp('LinChirps'),
disp('Chirps'),
disp('sineoneoverx'),
disp('Cusp2'),
disp('SmoothCusp'),
disp('Gabor'),
disp('Piece-Regular');
disp('Piece-Polynomial');
disp('Gaussian');
end

```

```

%
% Originally made by David L. Donoho.
% Function has been enhanced.

```

```

%
% Part of WaveLab Version 802
% Built Sunday, October 3, 1999 8:52:27 AM
% This is Copyrighted Material
% For Copying permissions see COPYING.m
% Comments? e-mail wavelab@stat.stanford.edu
%

```

denoisewl.m

```
%*****
% Denoise:
%   via orthogonal wavelet transform. We modify each
%   detail and approx.function by prediction
%   %
% SYNTAX: x_real=denoisewl(xn,p,nx)
%
% INPUT: xn = Received signal
%        p= predictor order
%        nx = number of scales used
%
% OUTPUT: x_real = Denoised signal
%
%
% SUB_FUNC: depred
% Written by ralph hippenstiel
%*****
%*****
```

```
function x_real=denoisewl(xn,p,nx);
```

```
[cx lx]=wavedec(xn,nx,'db4');
```

```
dxs=[];
for i=1:nx
    d=detcoef(cx,lx,i);
    dl=length(d);

    dc=depred(d,p);
    dxs=[dc dxs];
end
a=appcoef(cx,lx,'db4',nx);

ac=depred(a,p);
dxs=[ac dxs];

xd=waverec(dxs,lx,'db4');
x_real=xd;
```

dnswlmed.m

```
%*****
%*****
% Denoise:
%   via orthogonal wavelet transform. We modify each
%   detail and approx.function by prediction
%   %
% SYNTAX: x_real=dnswlmed(xn,nx)
%
% INPUT: xn = Received signal
%        nx = number of scales
%
%
% OUTPUT: x_real = Denoised signal
%
%
% SUB_FUNC: None
% Written by ralph hippenstiel
%*****
%*****
```

```
function x_real=dnswlmed(xn,nx);

% nx defines the level of decomposition

[cx lx]=wavedec(xn,nx,'db4');

dxc=[];
for i=1:nx
    d=detcoef(cx,lx,i);
    dl=length(d);

    dc=medfilt1(d);
    dxc=[dc dxc];
end
a=appcoef(cx,lx,'db4',nx);
ac=medfilt1(a);
dxc=[ac dxc];
xd=waverec(dxc,lx,'db4');
x_real=xd;
```

predi.m

```
%%%%%%%%%%
% denoises the signal using overlapped fft's and lin. prediction
% function x1_real=predi(x,p)
% p=predictor order
% output recy=denoised signal
% calls subroutine depred(t,p)
% ffts segments of the sequence using triangular windows
% written by Ralph Hippenstiel
%%%%%%%%%

function x_real=predi(x,p);

fx=[];

for i=1:61

    fx(i,:)=fft(x((i-1)*8+1:(i-1)*8+32).*triang(32));

end

for k=1:17
    t=fx(:,k);
    xh=depred(t,p);
    fx(:,k)=conj(xh)';
end
    fys=conj(fliplr(fx(:,2:16)));
fx(:,18:32)=fys;

%recover the filtered sequence
recy(512)=0;
for i=1:61
    recy((i-1)*8+1:(i-1)*8+32)=recy((i-1)*8+1:(i-1)*8+32)+ifft(fx(i,:));
end

x_real=recy*0.5;
```

denoise_med.m

```
%%%%%%%%%%  
%denoises using a 1D median filter along time  
%subroutine denoise_med(x)  
%written by Ralph Hippenstiel  
%%%%%%%%%
```

```
function x_real=denoise_med(x);
```

```
%transform (fft) the data
```

```
for i=1:61
```

```
    fx(i,:)=fft(x((i-1)*8+1:(i-1)*8+32).*triang(32)');
```

```
end
```

```
%filter the transformed data
```

```
for i=1:17
```

```
    fy(:,i)=medfilt1(real(fx(:,i)),3)+j*medfilt1(imag(fx(:,i)),3);
```

```
end
```

```
    fys=conj(fliplr(fy(:,2:16)));
```

```
fy(:,18:32)=fys;
```

```
%recover the filtered sequence
```

```
recy(512)=0;
```

```
for i=1:61
```

```
    recy((i-1)*8+1:(i-1)*8+32)=recy((i-1)*8+1:(i-1)*8+32)+ifft(fy(i,:));
```

```
end
```

```
recy=recy*0.5;
```

```
x_real=recy;
```

depred.m

```
%%%%%%%%%%
%denoises the test signal using a predictor of length= size
% written by Ralph Hippenstiel
%%%%%%%%%
function xh=dpred(dat,siz)
xw(512)=0;
sdat=dat;
ko=length(sdat);
no=siz;
cor=xcorr(sdat);
r(1:no+1)=cor(ko:ko+no);
a(no)=0;
xh(ko)=0;
xh(1:no)=sdat(1:no);
R=toeplitz(r(1:no),conj(r(1:no)));
ro=r(2:no+1);
a=inv(R)*conj(ro');
for i=no+1:ko
xw(no)=0;
for k=1:no
xw(k)=a(k)*sdat(i-k);
end
xh(i)=sum(xw);
end
```

INITIAL DISTRIBUTION LIST

	No. Copies
1. Defense Technical Information Center 8725 John J. Kingman Rd, STE 0944 Ft. Belvoir, VA 22060-6218	2
2. Dudley Knox Library, Code 52 Naval Postgraduate School 411 Dyer Road Monterey, CA 93943-5101	2
3. Research Office, Code 09 Naval Postgraduate School 589 Dyer Road Monterey, CA 93943-5138	1
4. Chairman, Code EC Department of Electrical and Computer Engineering Naval Postgraduate School 833 Dyer Road Monterey, CA 93943-5121	1
5. Professor R. Hippenstiel, Code EC/Hi Chair, Electrical Engineering Department Engineering Building University of Texas at Tyler 3900 University Blvd Tyler, TX 75799	2
6. Professor Dave Kreztmann, Code SPKD NRO Chair Naval Postgraduate School 833 Dyer Road Monterey, CA 93943-5121	1
7. Mr. Donald Rogers Room 41D00F 414675 Lee Road Chantilly, VA 20151-1708	5